C# 프로그래밍

- 12 주차 (1강)

SEJONG UNIVERSITY

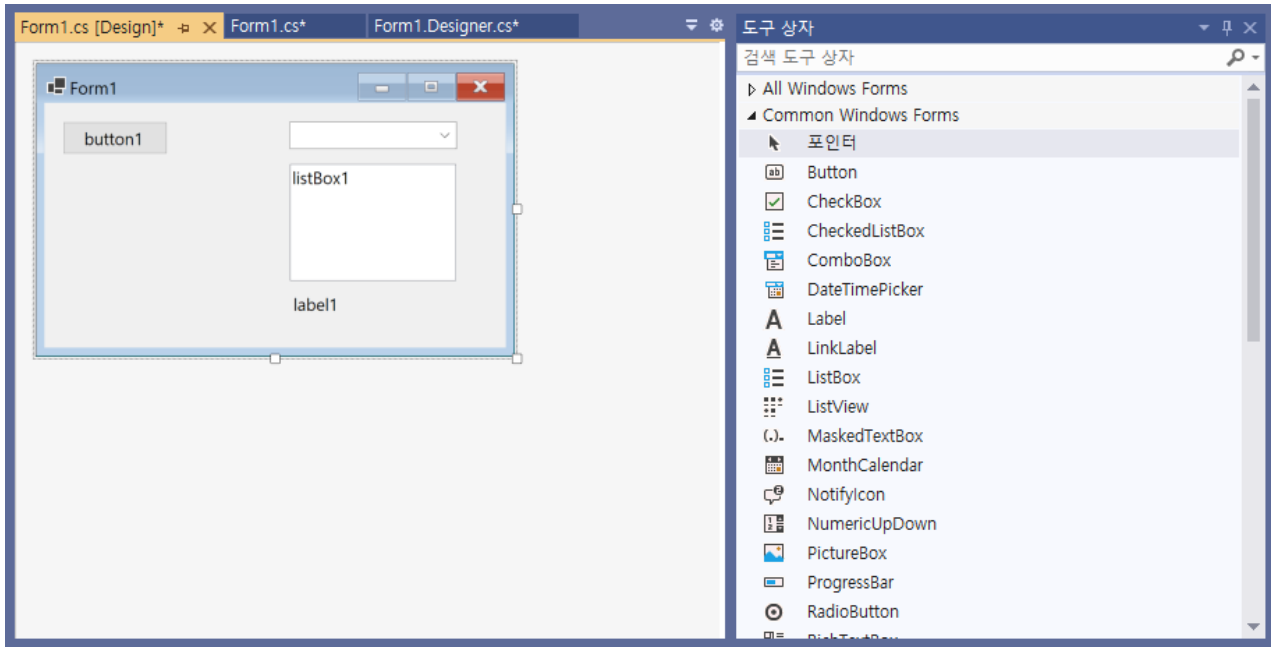
C# 코드로 WinForm 윈도우 생성

폼 디자이너를 이용한 WinForm UI 구성

SEJONG UNIVERSITY

WinForm 기반 사용자 인터페이스 (UI)

- .NET을 통한 사용자 인터페이스(UI) 생성
  - ✓ .NET 프레임워크 1.0부터 WinForm 제공
  - ✓ .NET 프레임워크 3.0부터 도입된 WPF(Windows Presentation Foundation) 이용
  - ✓ WPF가 더 다양한 효과를 제공하지만, 내용이 방대하여 익히기 어려움
- 비주얼스튜디오는 WinForm UI 구성을 위한 폼 디자이너 툴 제공
  - ✓ 개발자가 그림 그리듯 UI 생성 가능
  - ✓ 도구상자에서 버튼이나 콤보박스 등의 컨트롤을 끌어다 윈도우에 올려 놓음
  - ✓ 뒤로는 해당 UI에 대한 C# 코드를 자동으로 생성



C# 코드로 WinForm 윈도우 생성

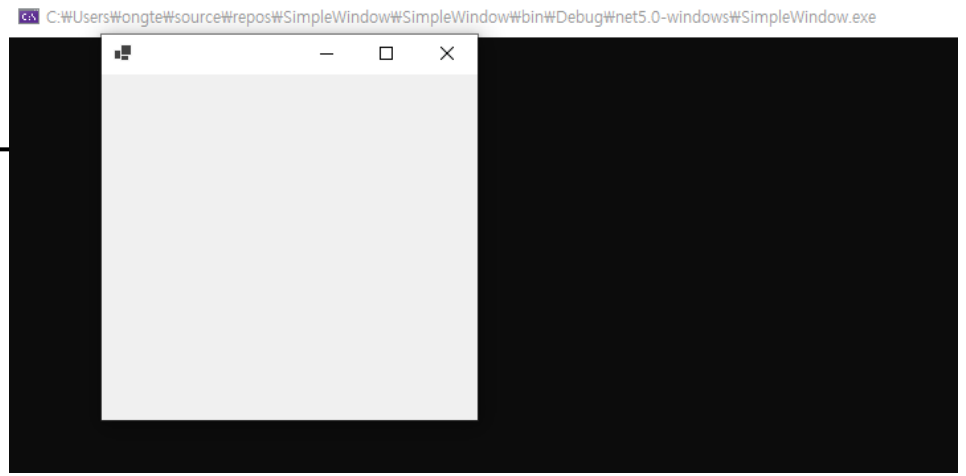
- .NET에서 WinForm 클래스를 이용한 윈도우 생성 절차
  - ✓ System.Windows.Forms.Form 클래스에서 파생된 윈도우 폼 클래스 선언
  - ✓ 생성한 클래스의 인스턴스를 System.Windows.Forms.Application.Run() 메소드에 인수로 넘겨 호출
- 비주얼스튜디오 프로젝트 생성 및 설정
  - ✓ 비주얼스튜디오를 실행하고 프로젝트 템플릿을 "콘솔 앱"으로 선택하여 생성
  - ✓ 솔루션 탐색기에서 생성한 프로젝트를 클릭하여 "프로젝트명.csproj"를 편집기로 읽음
  - ✓ 그림과 같이 파일 수정

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net5.0-windows</TargetFramework>
    <UseWindowsForms>true</UseWindowsForms>
    <DisableWinExeOutputInference>true</DisableWinExeOutputInference>
  </PropertyGroup>

</Project>
```

간단한 윈도우 생성 예제 코드

```
using System;

namespace SimpleWindow
{
    // MainApp이 System.Windows.Forms.Form 클래스로부터 상속받도록 선언
    class MainApp: System.Windows.Forms.Form
    {
        static void Main(string[] args)
        {
            // Application.Run() 메소드에 MainApp의 인스턴스를 인수로 넘겨 호출
            System.Windows.Forms.Application.Run(new MainApp());
        }
    }
}
```

실행 결과:



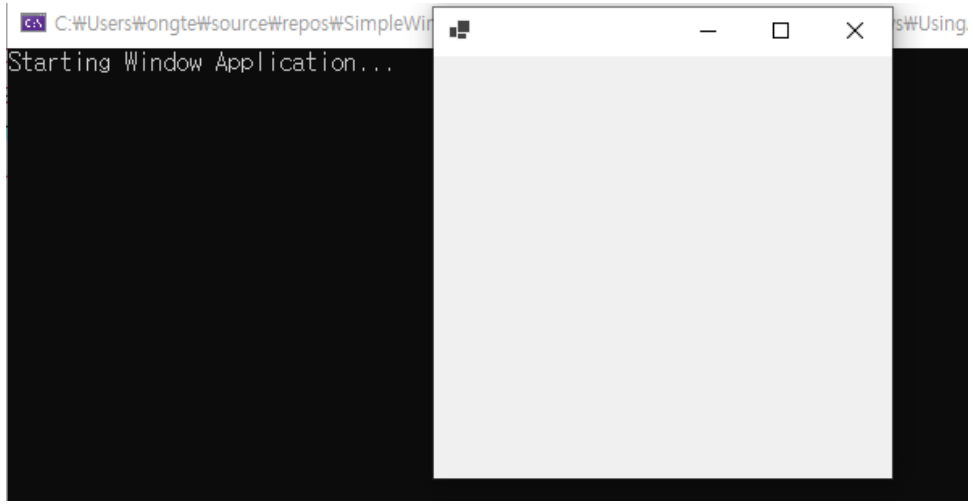
Application 클래스

- Application 클래스는 크게 두 가지 역할 수행
  - ✓ 윈도우 응용 프로그램을 시작하고 종료시키는 메소드 제공
  - ✓ 윈도우 메시지를 처리
- Application.Run( ) 메소드
  - ✓ 응용프로그램을 시작하는 메소드
- Application.Exit( ) 메소드
  - ✓ 해당 응용프로그램을 종료하는 메소드
  - ✓ 응용프로그램이 갖고 있는 모든 윈도우를 닫은 뒤 Run() 메소드가 반환되도록 함
  - ✓ Run() 메소드 뒤의 코드를 실행

윈도우 생성 및 종료 메소드 사용 예제 코드

```
using System;
using System.Windows.Forms;

namespace UsingApplication
{
    class MainApp : Form
    {
        static void Main(string[] args)
        {
            MainApp form = new MainApp();

            form.Click += new EventHandler(          // Form 클래스는 여러 가지 이벤트를 정의
                (sender, EventArgs) =>               // 그중 Click 이벤트는 윈도우를 클릭했을 때 발생
                {                                    // 미리 정의된 이벤트에 이벤트 처리기 메소드를 선언하여 등록
                    Console.WriteLine("Closing Window ...");
                    Application.Exit();
                });

            Console.WriteLine("Starting Window Application...");
            Application.Run(form);

            Console.WriteLine("Existing Window Application...");
        }
    }
}
```

실행 결과:



Microsoft Visual Studio 디버그 콘솔

```
Starting Window Application...
Closing Window ...
Existing Window Application...
```

Application 클래스 : 메시지 필터링

- 윈도우 기반 응용프로그램은 이벤트 기반 방식으로 동작
  - ✓ 마우스 클릭, 키보드 입력과 같이 갑자기 발생하는 사건(이벤트)에 반응하여 코드 실행
  - ✓ 마우스, 키보드를 제어하면 인터럽트 발생하고, 윈도우 OS가 인터럽트를 바탕으로 윈도우 메시지를 생성한 뒤 이벤트를 받아야하는 응용 프로그램에게 전달
- Application 클래스는 관심있는 메시지만 걸러낼 수 있는 메시지 필터링 기능 수행
  - ✓ 예를 들어, "Alt + F4" 키를 입력하여 응용프로그램을 종료하는 메시지를 걸러낼 수 있음
  - ✓ 윈도우 메시지는 식별 번호(ID)가 붙여져 있음
  - ✓ 특정 ID의 메시지를 걸려내는 필터 등록할 수 있음
- Application.AddMessageFilter() 메소드는 응용프로그램에 메시지 필터 설치
  - ✓ IMessageFilter 인터페이스를 구현하는 파생 클래스의 인스턴스를 인수로 받음

```
public interface IMessageFilter
{
    bool PreFilterMessage(ref Message m);
}
```

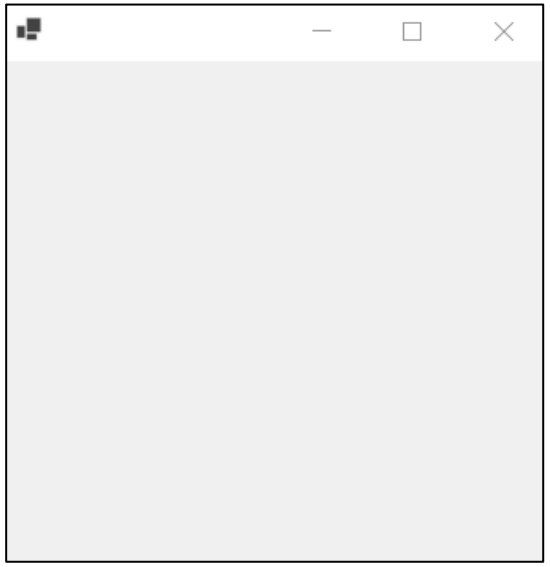
메시지 필터링 예제 코드

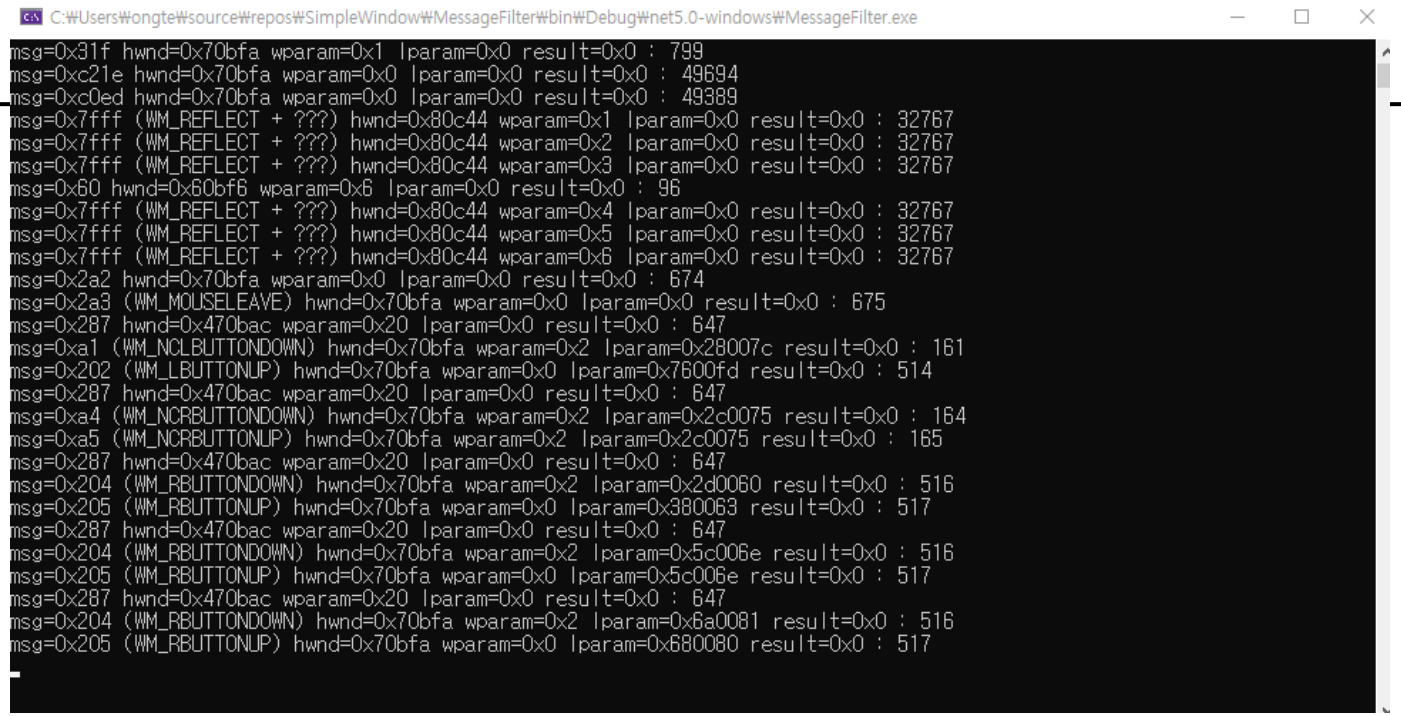
```
using System;
using System.Windows.Forms;

namespace MessageFilter
{
    class MessageFilter : IMessageFilter
    {
        public bool PreFilterMessage(ref Message m)
        {
            if (m.Msg == 0x0F || m.Msg == 0xA0 || // 0x0F: WM_PAINT, 0x200: WM_MOUSEMOVE 등
                m.Msg == 0x200 || m.Msg == 0x113)
                return false;  // 해당 메시지는 건드리지 않았음

            Console.WriteLine($"{m.ToString()} : {m.Msg}"); // 메시지 ID 출력

            if (m.Msg == 0x201) // 0x201 (윈도우 클릭) 메시지 처리
                Application.Exit();

            return true; // 해당 메시지는 처리했으니 응용 프로그램에서 관심가질 필요 없음
        }
    }
```

메시지 필터링 예제 코드

```
class MainApp : Form
    {
        static void Main(string[] args)
        {
            Application.AddMessageFilter(new MessageFilter()); // 메시지 필터 설치
            Application.Run(new MainApp());
        }
    }
}
```

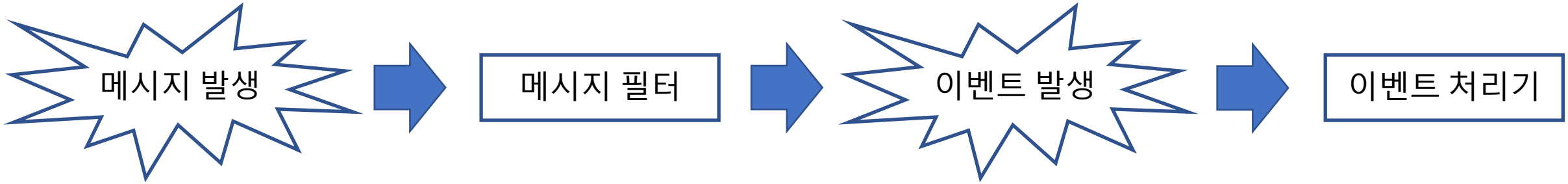
실행 결과:





윈도우를 표현하는 Form 클래스

- Form(과 컨트롤)에 정의된 이벤트와 이벤트 처리기 연결하기
- Form의 프로퍼티를 조절하여 윈도우 모양 바꾸기
- Form 위에 컨트롤 올리기



Form에 정의된 이벤트와 이벤트 처리기 연결

- Form 클래스는 운영체제가 보내는 메시지 중 일부에 대해 이벤트를 구현
  - ✓ 예를 들어, 윈도우 위에서 마우스 왼쪽 버튼을 클릭하면 WM_LBUTTONDOWN 메시지가 Form 객체에 전달
  - ✓ Form 객체는 이에 대해 MouseDown 이벤트를 발생
  - ✓ 미리 정의된 이벤트에 이벤트 처리기 메소드를 선언하여 등록 필요

사용예제:

```
class MyForm : Form
{
    // 이벤트 처리기 선언
    private void Form_MouseDown(object sender, MouseEventArgs e)
    {
        MessageBox.Show("안녕하세요!");
    }

    public MyForm()
    {
        // 이벤트 처리기를 이벤트에 연결
        this.MouseDown += new MouseEventHandler(this.Form_MouseDown);
    }
}
```

Form에 정의된 이벤트와 이벤트 처리기 연결

- MouseDown 이벤트의 선언

```
public event MouseEventHandler MouseDown;
```

✓ MouseEventHandler는 대리자 (이벤트는 대리자를 기반으로 선언)

```
public delegate void MouseEventHandler(object? sender, MouseEventArgs e);
```

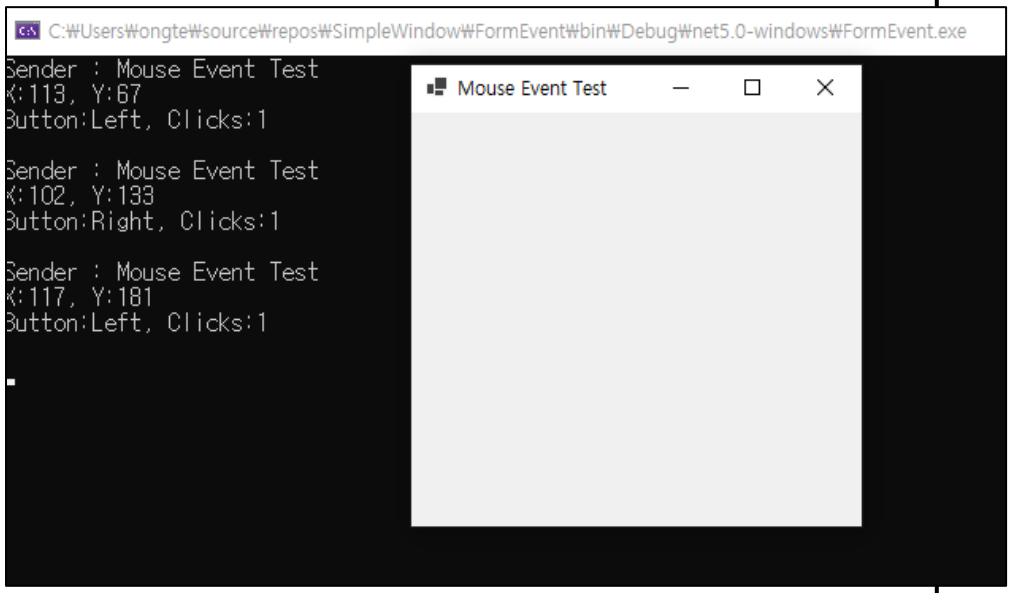
- sender: 이벤트가 발생한 객체 (예: Form 또는 Button 객체 자신)
- MouseEventArgs: 아래의 프로퍼티 제공하여 마우스 이벤트의 상세정보 제공

프로퍼티	설명
Button	마우스의 어떤 버튼에서 이벤트 발생했는지 나타냄
Clicks	마우스의 버튼 클릭한 횟수
Delta	마우스 휠의 회전 방향과 회전한 거리
X	마우스 이벤트가 발생한 폼 또는 컨트롤상의 x 좌표
Y	마우스 이벤트가 발생한 폼 또는 컨트롤상의 y 좌표

Form에 정의된 이벤트 처리 예제 코드

```
class MainApp : Form
{
    public void MyMouseHandler(object sender, MouseEventArgs e)
    {
        Console.WriteLine($"Sender : {((Form)sender).Text}");
        Console.WriteLine($"X:{e.X}, Y:{e.Y}");
        Console.WriteLine($"Button:{e.Button}, Clicks:{e.Clicks}");
        Console.WriteLine();
    }

    public MainApp(string title)
    {
        this.Text = title;
        this.MouseDown += new MouseEventHandler(MyMouseHandler);
    }

    static void Main(string[] args)
    {
        Application.Run(new MainApp("Mouse Event Test"));
    }
}
```

실행 결과:



Form 프로퍼티를 조절하여 윈도우 모양 바꾸기

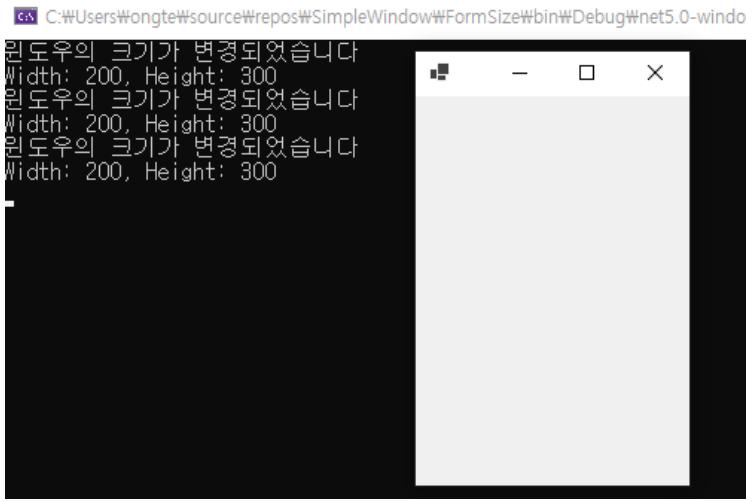
- Form 클래스는 윈도우 모양을 결정짓는 크기, 배경색, 전경색, 투명도, 제목, 폰트 등 여러가지 프로퍼티를 가지고 있음

```
using System;
using System.Windows.Forms;

namespace FormSize
{
    class MainApp : Form
    {
        static void Main(string[] args)
        {
            MainApp form = new MainApp();
            form.Width = 300;
            form.Height = 200;

            form.MouseDown += new MouseEventHandler(form_MouseDown);
            Application.Run(form);
        }
```

Form 프로퍼티를 조절하여 윈도우 모양 바꾸기

```
    static void form_MouseDown(object sender, MouseEventArgs e)
    {
        Form form = (Form)sender;
        int oldWidth = form.Width;
        int oldHeight = form.Height;

        if(e.Button == MouseButtons.Left)
        {
            if(oldWidth < oldHeight)
            {
                form.Width = oldHeight;
                form.Height = oldWidth;
            }
        }
        else if (e.Button == MouseButtons.Right)
        {
            if (oldHeight < oldWidth)
            {
                form.Width = oldHeight;
                form.Height = oldWidth;
            }
            Console.WriteLine("윈도우의 크기가 변경되었습니다");
            Console.WriteLine($"Width: {form.Width}, Height: {form.Height}");
        }
    }
}
```

실행 결과:



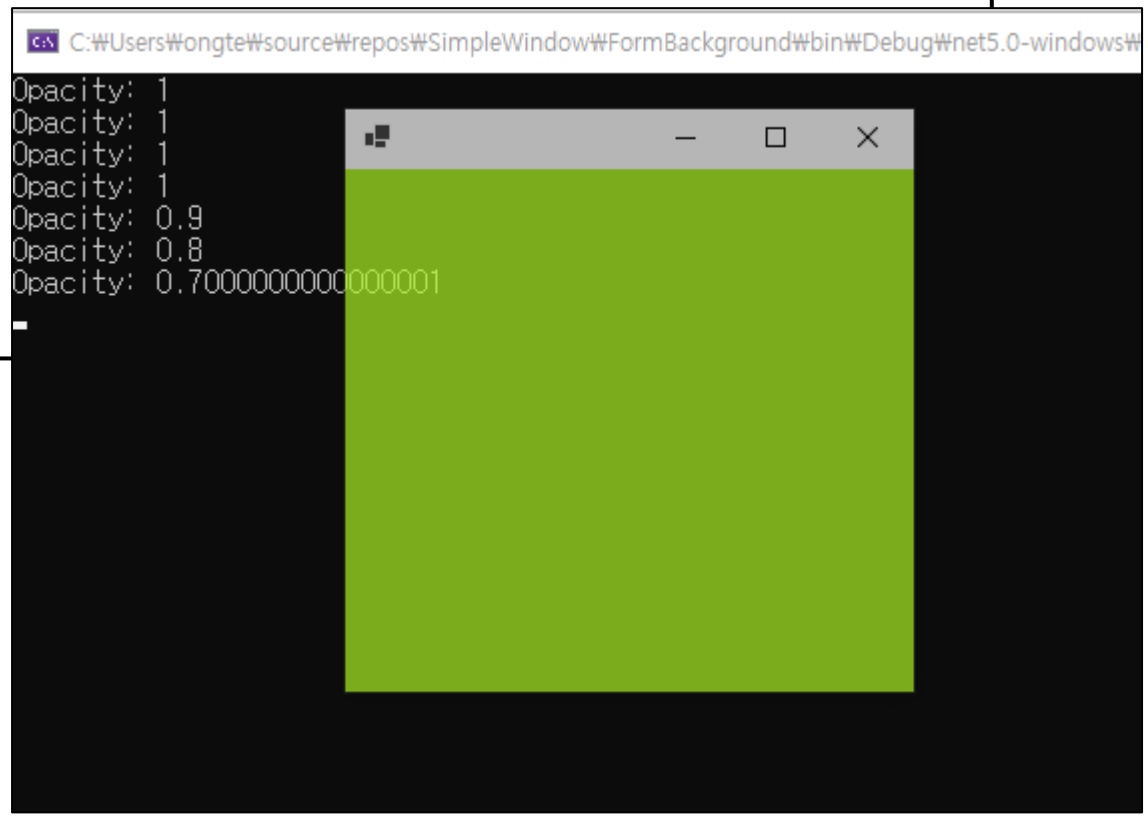
윈도우 배경색과 투명도 바꾸기

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace FormBackground
{
    class MainApp : Form
    {
        Random rand;
        public MainApp()
        {
            rand = new Random();
            this.MouseWheel += new MouseEventHandler(MainApp_MouseWheel);
            this.MouseDown += new MouseEventHandler(MainApp_MouseDown);
        }

        void MainApp_MouseDown(object sender, MouseEventArgs e)
        {
            if(e.Button == MouseButtons.Left)
            {
                Color oldColor = this.BackColor; // BackColor 프로퍼티 통해 윈도우 배경색 바꿀 수 있음
                this.BackColor = Color.FromArgb(rand.Next(0, 255), // Color 클래스의 정적메소드나 미리정의된
                                                rand.Next(0, 255), // 미리 정의된 상수값 이용
                                                rand.Next(0, 255));
            }
        }
```

윈도우 배경색과 투명도 바꾸기

```
 void MainApp_MouseWheel(object sender, MouseEventArgs e)
        {   // Opacity 프로퍼티 통해 윈도우 투명도 바꿀 수 있음
            // 0에 가까울 수록 투명, 1에 가까울 수록 불투명
            this.Opacity = this.Opacity + (e.Delta > 0 ? 0.1 : -0.1);
            Console.WriteLine($"Opacity: {this.Opacity}");
        }

        static void Main(string[] args)
        {
            Application.Run(new MainApp());
        }
    }
}
```

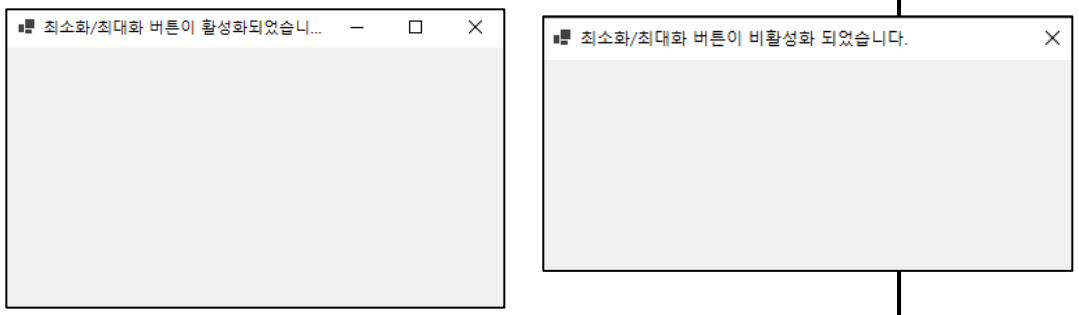
실행 결과:



최대화/최소화 버튼 바꾸기

```
class MainApp : Form
{
    static void Main(string[] args)
    {
        MainApp form = new MainApp();

        form.Width = 400;
        form.MouseDown += new MouseEventHandler(form_MouseDown);

        Application.Run(form);
    }
```

실행 결과:




```
    static void form_MouseDown(object sender, MouseEventArgs e)
    {
        Form form = (Form)sender;

        if(e.Button == MouseButtons.Left)
        {
            form.MaximizeBox = true; // 최대화 버튼 표시
            form.MinimizeBox = true; // 최소화 버튼 표시
            form.Text = "최소화/최대화 버튼이 활성화되었습니다";
        }
        else if(e.Button == MouseButtons.Right)
        {
            form.MaximizeBox = false; // 최대화 버튼 감추기
            form.MinimizeBox = false; // 최소화 버튼 감추기
            form.Text = "최소화/최대화 버튼이 비활성화 되었습니다.";
        }
    }
```

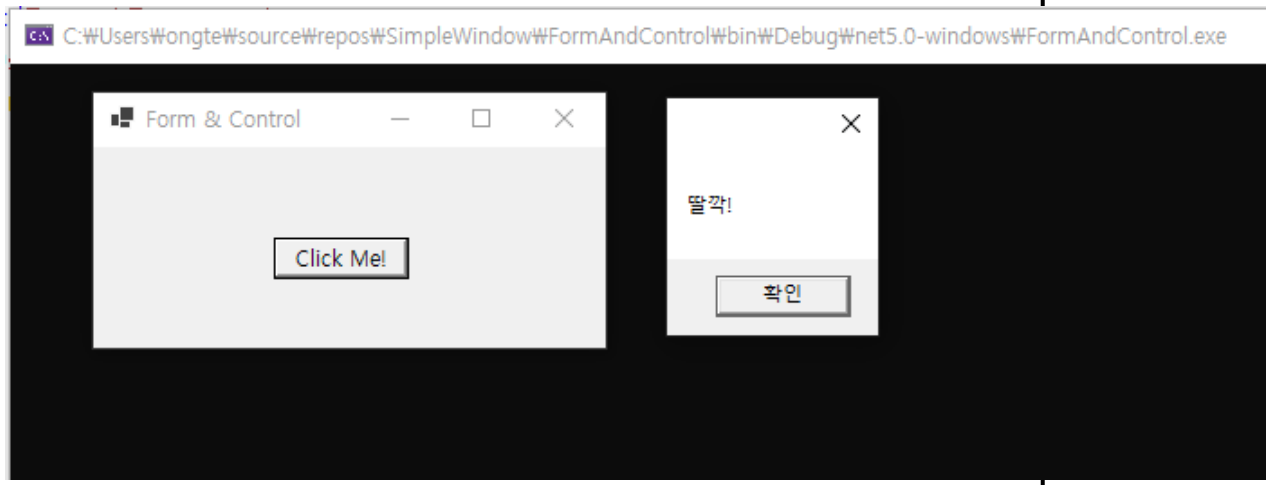
Form 위에 컨트롤 올리기

- 컨트롤이란?
  - ✓ 윈도우 OS가 제공하는 사용자 인터페이스 요소
  - ✓ 메뉴, 콤보박스, 리스트뷰, 버튼 텍스트박스 등
- 컨트롤을 폼 위에 올리기 위한 구현 과정
  - ✓ 컨트롤의 인스턴스 생성
  - ✓ 컨트롤의 프로퍼티에 값 지정
  - ✓ 컨트롤의 이벤트에 이벤트 처리기 등록
  - ✓ 폼에 컨트롤 추가

버튼을 가진 윈도우 예제 코드

```
class MainApp : Form
{
    static void Main(string[] args)
    {
        Button button = new Button(); // 컨트롤의 인스턴스 생성

        button.Text = "Click Me!"; // 컨트롤의 프로퍼티 값 지정
        button.Left = 100;
        button.Top = 50;

        button.Click +=    // 컨트롤의 이벤트에 이벤트 처리기 등록
            new EventHandler((object sender, EventArgs e) =>
            {
                MessageBox.Show("딸깍!");
            });

        MainApp form = new MainApp();
        form.Text = "Form & Control";
        form.Height = 150;

        form.Controls.Add(button); // 폼에 컨트롤 추가
        Application.Run(form);
    }
}
```

실행 결과:



Thank you!